#Task 1 - Directory Reporting Script

```
#Variables
$path = 'c:\temp'
$output = 'c:\temp\directory_report.txt'

#Function to list object(s)
Function listDir{
  Param(
    [ValidateNotNullorEmpty()]
    [Parameter(Mandatory=$true,ValueFromPipeline=$true)]
    $path,
    [ValidateNotNullorEmpty()]
    [Parameter(Mandatory=$true,ValueFromPipeline=$true)]
    $output
  )
  get-childitem -path $path |`
    select-object -property length,fullname,lastwritetime |`
      foreach {
        $timestamp = (get-date).toString('HH:mm:ss.fff')
        write-host "$($timestamp): name:$($_.fullname), size:$($_.length), last Modified: $($_.LastWriteTime)"
        add-content $output -value "$($timestamp): name:$($_.fullname), size:$($_.length), last Modified: $.  ($_.LastWriteTime)"
    }
}

#Begin
listDir -path $path -output $output
#EOF
```

#Task 2 - System Patching Script

```
#Variables
$output = 'c:\temp\patch_log.txt'

#Begin
#Check and list available update(s)
$UpdateSession = New-Object -ComObject "Microsoft.Update.Session"
```

```
$UpdateSearcher = $UpdateSession.CreateupdateSearcher()
$Updates = @($UpdateSearcher.Search("IsHidden=0 and IsInstalled=0").Updates)

#Add update(s) to array
$objCollection = New-Object -ComObject "Microsoft.Update.UpdateColl"
Foreach($update in $updates){
  $objCollection.add($update)
  $timestamp = (get-date).toString('HH:mm:ss.fff')
  write-host "Title: $($$update.Title), KB: $($Update.KBArticleIDs), Size: $([System.Math]::Round($Update.MaxDownloadSize/1MB,2))"
  Add-content $output "$($timestamp): Title: $($$update.Title), KB: $($Update.KBArticleIDs), Size:
$([System.Math]::Round($Update.MaxDownloadSize/1MB,2))"
}

If(!$update){
  $timestamp = (get-date).toString('HH:mm:ss.fff')
  write-host "No update(s) were found at this time." -ForegroundColor Green
  Add-content $output "$($timestamp): No update(s) were found at this time."
}else{
  #Download available update(s)
  $Downloader = $UpdateSession.CreateUpdateDownloader()
  $Downloader.Updates = $objCollection
  Try{
    $DownloadResult = $Downloader.Download()
    $timestamp = (get-date).toString('HH:mm:ss.fff')
    Write-Host "Update(s) successfully downloaded." -ForegroundColor Green
    Add-content $output "$($timestamp): Update(s) successfully downloaded."
  }Catch{
    $timestamp = (get-date).toString('HH:mm:ss.fff')
    Write-Host "Error while downloading update(s), $($Error[0].Exception.Message)" -ForegroundColor Yellow
    Add-content $output "$($timestamp): Error while downloading update(s), $($Error[0].Exception.Message)"
  }

  #Install available update(s)
  if($DownloadResult.ResultCode -eq 2){
    $objInstaller = $updateSession.CreateUpdateInstaller()
    $objInstaller.Updates = $objCollection
```

```
      try{
        $InstallResult = $objInstaller.Install()
        $timestamp = (get-date).toString('HH:mm:ss.fff')
        write-host "Update(s) successfully installed." -ForegroundColor Green
        Add-content $output "$($timestamp): Update(s) successfully installed."
      }Catch{
        $timestamp = (get-date).toString('HH:mm:ss.fff')
        write-host "Error while installing update(s)." -ForegroundColor Yellow
        Add-content $output "$($timestamp): Error while installing update(s)."
      }
  }
}
#EOF
```

#Task 3 - Reset and Disable Local Administrator/Root Account

```
#Variables
$userName = 'administrator'
$newPass = ConvertTo-SecureString -AsPlainText 'Test@!NinjaOne' -force
$output = 'c:\temp\account_management_log.txt'

#Begin
#Reset account password
Try{
  Set-LocalUser $username -password $newPass -errorAction stop
  $timestamp = (get-date).toString('HH:mm:ss.fff')
  write-host "$($timestamp): The user account '$($userName)' password has been redefined."
  add-content $output "$($timestamp): The user account '$($userName)' password has been redefined."
}Catch{
  $timestamp = (get-date).toString('HH:mm:ss.fff')
  write-host "$($timestamp): Error while redefining the user account '$($userName)' password."
  add-content $output "$($timestamp): Error while redefining the user account '$($userName)' password."
}

#Disable user account
Try{
  Get-LocalUser $username | Disable-LocalUser -errorAction stop
  $timestamp = (get-date).toString('HH:mm:ss.fff')
  write-host "$($timestamp): The user account '$($userName)' has been successfully disabled."
  add-content $output "$($timestamp): The user account '$($userName)' has been successfully disabled."
}Catch{
  $timestamp = (get-date).toString('HH:mm:ss.fff')
  write-host "$($timestamp): Error while disabling the user account '$($userName)'."
  add-content $output "$($timestamp): Error while disabling the user account '$($userName)'."

}
#EOF
```